Reusable Design Processes via Modular, Executable, Decision-Centric Templates

Jitesh H. Panchal[*], Marco Gero Fernández[§], Christiaan J. J. Paredis[#] and Farrokh Mistree[**]
Systems Realization Laboratory, The George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0405

While there have been many advances with respect to reusability and scalability of product architectures over the past several decades, little progress has been made in applying the same concepts to underlying design processes. It is on this aspect of design process design that we focus in this paper. Design processes play a key role in product design and their configuration has a significant effect on both the efficiency and the effectiveness with which resources are committed. Design processes also directly influence the final design of the product under consideration. As such, more attention must be paid to the manner in which these processes are modeled so that they may be standardized, executed, analyzed, and stored, allowing for their leveraging across product lines and reducing product development times. Computer interpretability is a key consideration in making required adjustments as product considerations evolve and design requirements change from one product to the next. In this paper, we offer a fundamental step in this direction by presenting a method for modeling design processes as reusable process templates that can be captured, archived, analyzed and manipulated on a computer.

I. Frame of Reference

A. Design Process Reuse

HOW similar do two (or more) products have to be in order to reuse the processes underlying their design? The answer varies depending on the level of abstraction at which the processes are modeled. For example, the Pahl and Beitz[1] design process is widely applicable to almost any mechanical design problem. However, at a computational level, were the design process is defined as a series of computational operations, the reusability of design processes, thus far, is extremely limited. It is reusability at this level that we seek to improve.

Consider a simple example, involving the design of two commonly employed mechanical components, namely, a pressure vessel and a spring, pictured in Figure 1. While both of these products can be described in terms of the geometric constraints, describing their form, and mechanical relations describing their function, they are nevertheless fundamentally different – with regard to the design parameters describing form, function, and behavior. Hence, computational design processes are problem specific and cannot be directly leveraged from one problem to another.

When considering the design processes underlying the products in Figure 1, however, there are certain similarities that emerge. Each design process can be considered to be a sequence of decisions and tasks. It is in terms of these information transformations that we develop a generic design process model, that can be executed, analyzed stored, and reused, regardless of context, engineering domain, or scale of the product considered. The required "genericism" of the underlying process model is achieved via a separation of the *declarative* (i.e., problem specific information) from the *procedural* (i.e., process specific information) flows of information. It is at the hand of the spring and pressure vessel design examples that we seek to illustrate relevant concepts. Although these examples are rather simple in nature, they nevertheless constitute a convenient means of illustrating the novelty of our method.

[*] Graduate Research Assistant, AIAA Student Member
[§] NSF IGERT Fellow, AIAA Student Member
[#] Assistant Professor
[**] Professor and Corresponding Author, AIAA Associate Fellow. Email: farrokh.mistree@me.gatech.edu. Phone: (404) 894-8412 Fax: (404) 894-9342

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2004		**00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Reusable Design Processes via Modular, Executable, Decision-Centric Templates	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Georgia Institute of Technology,The George W. Woodruff School of Mechanical Engineering,Systems Realization Laboratory,Atlanta,GA,30332**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**13**	

Before reviewing current process modeling techniques in Section II, however, we elucidate the needs for pursuing template-based design process modeling and establish the requirements for developing such an approach in Section I.B.

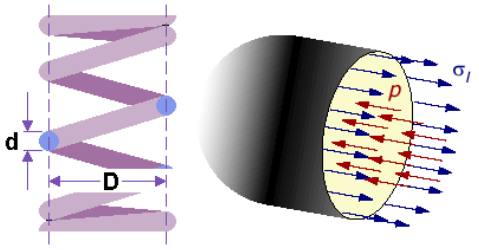


Figure 1. Helical spring and pressure vessel.

B. Modeling Design Processes as Templates

One of the main challenges in modeling any design effort, regardless of scale or scope, is formalizing the manner in which information flows and dependencies are represented. Another challenge lies in representing design processes in a domain neutral form that supports designers in providing and structuring required information content. This calls for a domain independent means of capturing design processes in an archivable and executable manner. It is for this reason that we advocate a template-based approach to modeling design processes. A *template* is commonly defined as (1) a pattern, used as a guide in making something accurately (2) a document or file having a preset format, used as a starting point for a particular application so that the format does not have to be recreated each time it is used.* Clearly, the word *template* is appropriate in our context because it implies reusability, achievability, and support/guidance.

In order to effectively support engineering design processes, this notion translates to the development of reusable computational models that can serve as building blocks. Such building blocks must also facilitate analysis, and execution. Currently, there is a lack of formal, executable, computational models for representing and reusing existing knowledge about design processes. The only knowledge that is readily available is confined either to designers' expertise or to descriptive/pictorial forms of documentation. This is a result of the predominantly narrative or symbolic nature of current models.

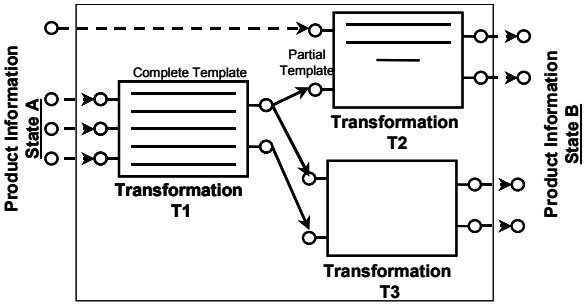


Figure 2. Design Process Modeled as Templates.

In order to address these challenges, we propose the use of domain independent design process templates. The design process templates result from a combination of templates for commonly encountered information transformations and the interactions between these transformations. The process templates are defined as computer-based representations of information transformations with well-defined inputs and outputs. The resulting design process templates, analogously to the building block templates from which they are composed, can be executed, stored, analyzed, and reused. The concept of constructing process templates from networks of design process building blocks is illustrated in Figure 2. The design process in this figure involves three information transformations, namely, T1, T2, and T3. Each of these templates is at a different level of completion. T1 is a complete template, implying that all the information required for its execution is available. T3 on the other hand has yet to be instantiated relevant to the problem at hand and consequently, does not differ from a generic information transformation on which it is based.

We proceed to illustrate the shortcomings of current approaches to modeling engineering design processes at the hand of the pressure vessel and spring design examples in Section II, exhibit the details of our design process modeling technique in Section III, and demonstrate the separation of *declarative* and *procedural* information in Section IV. In Section V, we establish confidence in the capability of our approach to handle the intricacies of significantly more comprehensive design problems (e.g., involving further complications due to distribution, complexity, collaboration, multi-functionality, etc.), and illustrate its use at the hand of designing Linear Cellular Alloys. With this in mind, we review various efforts towards modeling processes currently underway.

II. Existing Literature on Design Process Modeling

Processes can be represented at various levels of detail, depending on the intended use of the resulting models. Most of the traditional process modeling methods like the Program Evaluation and Review Technique (PERT),[2,3] Gantt Charts,[3] IDEF 0,[4] etc. capture information at the activity level. As such, these tools are useful for making organizational decisions with regard to processes such as time utilization, resource allocation, task precedence,

* Compiled from *www.dictionary.com*

material flow, etc. Example applications of tools such as these include modeling manufacturing processes to study process characteristics including time scheduling, material processing, assembly/disassembly and packaging. In a collaborative design scenario, models of processes are needed for understanding and coordinating collaborative work, thereby defining conflict management.[5]

What are the challenges in modeling design processes? Design processes for mechanical systems are complex often due to the inherent complexity of the product itself. Interactions and iterations between various activities and stakeholders add to the complexity of product realization processes. Whitney[6] points out that the complexity of mechanical designs results from the multifunctional nature of the parts required to obtain efficient designs. The underlying design processes involve many organizational units and engineering disciplines and the level of human intervention comprises an additional barrier to process modeling. Modeling of design processes is also complex because these cannot be completely described *a priori*. Downstream activities are very much dependent on the information generated by upstream activities and the associated level of uncertainty is consistently high.

Various methods have been developed in order to model design processes. These methods can be categorized by the manner in which design is viewed as a process. Some of these views characterize design processes as series of activities, decisions, evolving functions, sets of transformations, search processes, etc. The resulting representation of design processes, of course, is directly dependent on the view of design process chosen. A summary of the approaches to design process modeling is provided in Table 1, a discussion follows.

Table 1. Process modeling efforts in the design literature

Process Modeling Effort	***View of Design***	***Modeling, analysis objective***	***Basic units of a process***
IDEF [4]	Activity based	Organizational decisions	Activities, information
DSM[7]	Activity/Task based	Organizational decisions, risk, complexity, probability of rework, iterations, etc.	Tasks
Shimomura[8]	Functional Evolution	Capture design process, designers' intentions, trace design processes	Functional realization, functional operation, functional evaluation
Ullman [9]	Evolution of product states	Process representation	Abstraction, refinement, decomposition, patching combination, combination
Maimon[10]	Knowledge Manipulation through ASE	Development of a mathematical theory for design	Artifact space, specs, Analysis, synthesis, evaluation
Maher[11]	Knowledge manipulation	Development of knowledge based systems	Decomposition, case based reasoning, transformation
Gorti[12]		Development of engineering knowledge base	Goal, plan, specification, decision and context
DSP Technique[13]	Decision Based Design	Modeling, analyzing, debugging, finding inconsistencies in a design process	Phases, events, decisions, tasks, information

1. Activity-Based View of Design Processes

The design process, when viewed as a set of activities, can be subjected to organizational or scheduling analysis. Graph-based and matrix-based methods can be used to represent these processes. The graph-based techniques use Activity-Net-based models. Activity-Net-based models[14] are the earliest and most widely used techniques for modeling processes. These activity-net-based models are instantiated in two distinct representations, namely Activity-on-Node (AoN) and Activity-on-Arc (AoA). AoN representations are more applicable when the precedence of activities is known, whereas AoA representations are used when graphical identification and visualization of process events is important. Each of these models is thus used as a means of analyzing and comparing the complexity of processes -- performing risk-based analysis on aspects such as the expected time required for different tasks and obtaining a critical path. The design structure matrix (DSM)[7] is a popular means for representing both products and processes. Through DSM, hierarchical structures of both products and processes can be captured. The main advantage of using DSM is the ability to identify both interactions and iterations in a design process. Browning

and Eppinger[15] use DSM to model processes as sets of activities and process architectures as processes, along with their patterns of interaction. DSM is used for a variety of analyses including cost, schedule, risk tradeoff, probability of rework, level of interaction, complexity, iteration, and process improvement.

2. Functional-Evolution-Based View of Design Processes

Shimomura and co-authors[8] portray design as a process of functional evolution. Design is represented as a process in which a representation of a design object, which includes function, is gradually refined over time. The representation of the design object is based on the Function-Behavior-Structure (FBS)[16] model. Each functional evolution involves functional realization (i.e., converting function into structure), functional evaluation (i.e., confirming functional description with behavior) and functional operation (i.e., adding functional elements and functional relations to functional description). The authors present functional content as a measure of functional satisfaction. One of the advantages of this technique is the ability to model the product (as an FBS model) and process (as functional evolution) in an integrated fashion. This model can be used to both trace the design process and capture design intent.

3. Product-State-Evolution-Based View of Design Processes

A view similar to the functional evolution is the evolution of product states.[17] In this view, the design process is considered to be a problem solving technique centered on dynamically moving around a so called *product state space*. A product state represents all the information describing the product at a given point in the design process. Tomiyama, Yoshikawa, and co-authors[18,19] view design as a mapping of a point in the *function space* onto a point in the *attribute space*. Ullman[9] has also viewed the process of design as the refinement of a design from its *initial state* to its *final state*. According to Ullman, the essential components for aptly characterizing design processes are: the plan, the processing action, the effect, and the failure action. The various effects of a design process on the artifact are categorized as abstraction, refinement, decomposition, combination, combination, and patching.

Maimon and Braha[10] present the use of the Analysis-Synthesis-Evaluation (ASE) paradigm for representing design processes in terms of knowledge manipulation. Design processes are represented as tuples containing the artifact space, specifications, and transformation operators: analysis, synthesis, and evaluation (ASE). Zeng and Gu[20] also use models similar to ASE for developing a mathematical model of the design process. The authors develop a basic mathematical representation scheme to define objects involving the entire design process and investigate design processes via their mathematical representation. The elements of the design process proposed by Zeng and Gu include the following processes: synthesis and evaluation, design problem redefinition, and design decomposition.

4. Knowledge-Manipulation-Based View of Design Processes

Another effort focused on formalizing the representation of design knowledge within the design processes is purported by Maher.[11] Maher presents three models for knowledge representation: decomposition, case-based reasoning, and transformation. The focus of this work is on design synthesis for developing knowledge-based systems. Decomposition involves dividing large complex systems into smaller, less complex subsystems. Case-based reasoning involves generating design solutions from previous design problems. In transformation, available design knowledge is then expressed as a set of general transformational rules that can be used in a variety of scenarios.

5. Decision-Based View of Design Processes

Decision-based design (DBD) is a view of design that has been widely adapted for modeling design processes. Mistree and co-authors[21] view design as a process of converting information into knowledge about the product and decisions are the key markers used to determine the progress of design. Design processes can thus be modeled as sets of decisions. The Decision Support Problem (DSP) Technique[13,21-25] is a framework for design, developed based on this mindset. The DSP Technique[22] palette contains entities for modeling design processes, and allows for the arrangement and rearrangement of procedures or activities essential to design. The entities in the palette are used to build hierarchies and model design processes independent of the domain of the design under consideration.[21] The entities considered within the palette (i.e., tasks, decisions, events, and phases) are used to transform information from one state to another. Key decision types in engineering are also identified within the DSP Technique. These are *selection*[26,27], *compromise*[28,29], and combinations thereof (i.e., *coupled* decision). These decisions serve as the backbone for modeling design processes. In order to generate information required for executing decisions, supporting tasks are performed.

6. Representations of Design Processes

The Process Specification Language (PSL)[30] is an effort pursued by the National Institute of Standards and Technology (NIST) aimed at standardizing the representation of discrete processes (i.e., processes described as individually distinct events such as production scheduling, process planning, workflow, business process re-engineering, project management, etc). Gorti and co-authors[12] have developed object-oriented representations for design processes and products. The key elements of a design process, as identified by the authors, are goals, plans, specifications, decisions and context. The design artifact includes function, behavior, structure, and causal knowledge, relating objects to physical phenomena. Since the primary objective of the authors was to develop comprehensive engineering knowledge-bases, they did not focus on design process analysis.

Summary of Design Process Models

Each of these efforts towards modeling design processes are summarized in Table 1. While some of the methods are focused on capturing processes to make organizational decisions others focus on understanding and capturing designers' intentions. Even others center on the applicability of artificial intelligence. The most important realization is that there is a tradeoff between the broadness of model applicability, the granularity of information that can be represented, and the variety of analyses that can be performed using each of the models considered. For example, PERT, Gantt Charts, IDEF0, and Activity-Net-based models are very general in terms of applicability, but can be used to represent information only in terms of required activities and time. The kind of information being processed is not captured in any of these models, however. Due to these limitations, it has thus far not been possible to model design processes at a level sufficient for (1) execution, (2) analysis, and (3) reusability, as required for effective design process design. It is on addressing each of these three aspects in a consistent manner that we focus in this paper.

III. Systems-Based Modeling of Decision-Centric Design Processes

Our design process modeling strategy is based on two key assumptions: (1) design is a decision-centric activity and (2) design processes themselves are hierarchical systems. From a decision-centric standpoint, designing is a process of converting information that characterizes the needs and requirements for a product into knowledge about the product.[31,32] From the requirements to the final product, design processes are carried out through a number of phases. For example, the phases associated with Pahl and Beitz[33] design process are - planning and clarification of task, conceptual design, embodiment design and detailed design. Each phase is associated with *stages* of product information and converts information from one *stage* to another. Within each phase, there is a network of transformations that operate on product information. These transformations can be carried out in a sequential (as shown in Figure 3) or parallel fashion (not shown). The transformations operate on product related information and convert this information from one *state* to another. The state of information refers to the *amount* and *form* of that information that is available for design decision-making. For example, *analysis* is a transformation that maps the product form to behavior, whereas, *synthesis* is a mapping from expected behavior to the product form. It is important to note that these transformations remain same during different phases of the product realization process, as shown in Figure 3.

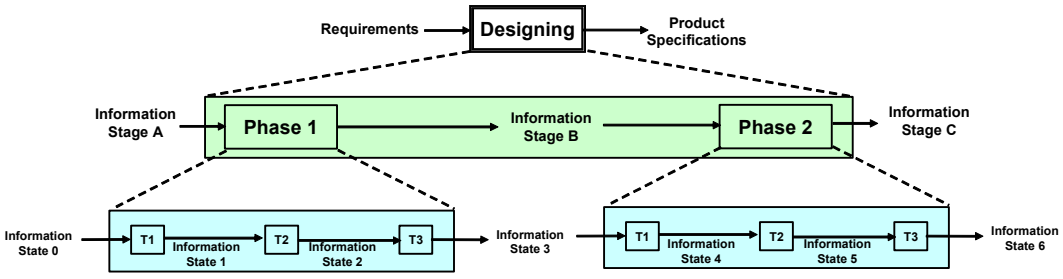


Figure 3. Sequential Design Process as a Series of Transformations.

From a hierarchical systems standpoint, design processes can be progressively broken down into sub-processes that can be further represented in terms of basic design process building blocks, namely the information transformations, discussed in the previous paragraph. Specifically, we focus on developing modular, reusable models of information transformations with clearly defined inputs and outputs that facilitate hierarchical modeling

of design processes. The design processes, modeled in such a manner, provide the ability to easily archive and reuse design process knowledge.

The design process model presented in this paper is an extension of the constructs developed within the DSP Technique proposed by Mistree and co-authors[22,25,31,32], rooted in the work of Simon.[34,35] The DSP Technique consists of three principal components: a design philosophy rooted in systems thinking, an approach to identifying and solving Decision Support Problems (DSPs), and software. 'Systems thinking' encourages designers to view products and processes as systems interacting with the environment. In the DSP Technique, support for human judgment in designing is offered through the formulation and solution of DSPs, which provide a means for modeling decisions encountered in design. The DSP Technique allows designers to model design processes at various levels of abstraction.[32] The level of required software support is different at different levels of abstraction.

As a part of the DSP Technique, a palette for modeling design processes using various entities such as phases, events, decision, tasks, and the system was developed.[22] Since there is a support problem associated with each DSP Technique palette entity, the use and reuse of design process and subprocess models, created and stored by others, is thus facilitated. Due to the domain independence of the underlying constructs and the overarching systems perspective, the DSP Technique offers a solid foundation for developing computational models of reusable design processes, as envisioned in this paper.

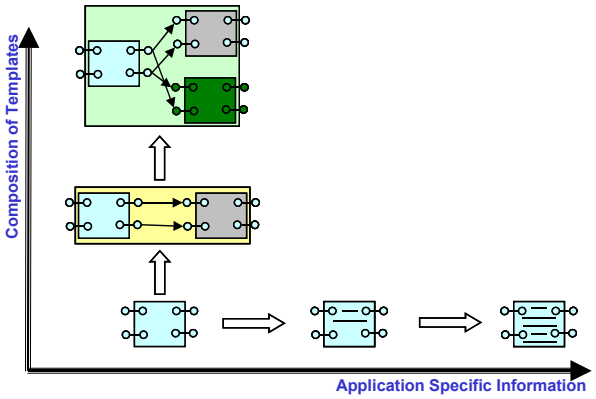


Figure 4. Reusability of design processes with regard to hierarchical composition and cross-domain application.

In the resulting model, the design processes are viewed as network of information transformations, as indicated in Figure 3. Our focus lies in developing generic constructs for modeling the most fundamental information transformations encountered in engineering design, including abstraction, composition, decomposition, interface, mapping, and synthesis. It is our intention to develop the corresponding support problems structured according to the overarching systems model envisioned in the DSP Technique. These information transformations are examples of tasks essential to supporting required design decisions. Since design tasks generate the information upon which the design decisions are based, our approach is decision-centric. Modeling a design process using such a decision centric approach involves developing networks of transformations with information-based interfaces.

In order to facilitate reuse of design process models, the building blocks of design processes must be generic. This requires modularity and domain independence. We aim to facilitate design process reuse with respect to (1) hierarchical composition and (2) cross-domain application, respectively. The underlying relationship between these two dimensions is illustrated in Figure 4. The domain independence of the underlying constructs is derived from the domain independence of underlying DSP Technique. The hierarchical composition (see Section V) is derived from the novel application of modularity to processes introduced in Section IV. The underlying concept centers on a separation of the declarative and procedural aspects of design processes, resulting in generic information transformation constructs that are instantiated as software templates. In fact, it is the nature of the information content, captured within the template that serves as the only differentiator among instantiated constructs, the underlying structure remaining the same.

IV. Design Process Reusability: Separation of Declarative Information from Procedural Information

In order to facilitate reusability of design processes across different design problems, the information about design processes is segmented into three layers as shown in Figure 5. These layers are discussed in detail in Sections IV.A through IV.C and include the product information layer, the declarative process layer, and the execution layer, respectively.

A. Product Information Level (Declarative Product Level)

In the layer corresponding to the product information level, only information, specific to the product being designed, is captured. Since this information is treated in a standardized manner, it can be used by different design processes. For example, the information associated with the design of either the spring or the pressure vessel, introduced in Section I, can be categorized as constituting design variables, responses, parameters, constraints,

goals, preferences, objectives, or analyses, as illustrated in the compromise DSP formulations in Table 2. We note that the two problems are notably different with dissimilar variables and relationships between them. The goals and constraints are also different. Although the product specific information is different, the inherent structure of the manner in which the information is used remains the same. Hence, it is possible to standardize the structure of information so that the creation of generic process elements becomes possible. The product information corresponding to these generic process elements for both the pressure vessel design and the spring design example are provided in Figure 6.

The process modeling technique proposed in this paper is analogous to architecture of a printed wiring board with a number of electronic components, such as those shown in Figure 7. The *wiring* corresponds to the flow of information in a process and the declarative process specific information discussed in Section IV.B is thus "hardwired". The *chips* plugged into the board define the manner in which the information is actually processed. Consequently, these chips correspond to the declarative (product specific) information, discussed in this section. A prime benefit is that the resulting reusability extends to both the chips and the board independently. Since procedural elements of information transformations are captured in a template form that is independent of the declarative aspects (i.e., the specific information considered), all aspects of information transformations from the components to the underlying interactions (represented by the "chips" and "wiring" in Figure 7, respectively) become modular. Both re-usability and reconfigure-ability are thus achieved.

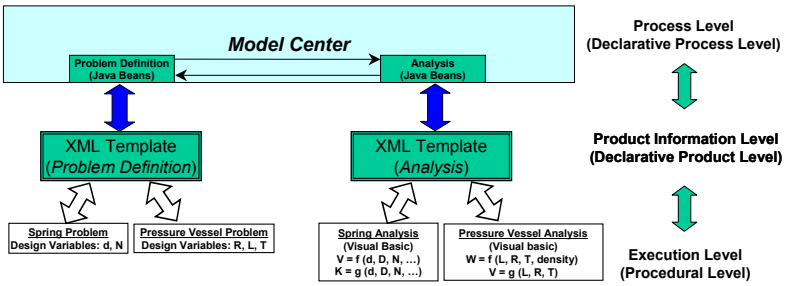


Figure 5 - Architecture of process modeling framework

Table 2. Compromise DSP formulations for pressure vessel and spring design.

Pressure Vessel Design	*Spring Design*
Given *Strength (S_t), Pressure (P), Density(ρ)* *Some helpful relations:* Volume, $V = \pi\left[\frac{4}{3}R^3 + R^2L\right]$ Weight, $W = \pi\rho\left[\frac{4}{3}(R+T)^3 + (R+T)^2L - (\frac{4}{3}R^3 + R^2L)\right]$ **Find** *System variables:* Radius (R) Length (L) Thickness (T) *Values of Deviation Variables:* *d1- (for weight goal)* *d2- (for volume goal)* **Satisfy** *System constraints:* $S_t - \left(\frac{PR}{T}\right) \geq 0$ $R - 5T \geq 0$ $(40 - R - T) \geq 0$ $(150 - L - 2R - 2T) \geq 0$ *System Goals (Normalized):* $d^-_{Volume} = 1 - \frac{V_{achieved}}{V_{target}}$ $d^-_{Weight} = 1 - \frac{W_{target}}{W_{achieved}}$ *Bounds on System Variables:* $0.1 \leq R \leq 36$ $0.1 \leq L \leq 140$ $0.5 \leq T \leq 6$ **Minimize** *Deviation Function:* $Z = w_1 d_1^- + w_2 d_2^-$	**Given** *Assumptions:* *Some helpful relations:* Deflection of spring: $\delta = \frac{8FD^3N}{d^4G}$ Solid height of spring: H=Nd, H≤0.5 in Stiffness of spring: $k = \frac{d^4G}{8D^3N}$ Volume of spring: $V = 0.25\pi^2 Dd^2(N+2)$ **Find** *System variables:* Wire diameter (d), Number of coils (N) *Values of Deviation Variables: d+, d- for goals* **Satisfy** *System constraints:* $6.957x10^{-6}\frac{N}{d^4} \geq 1.1$ $Nd \leq 0.5$ *System Goals (Normalized):* $53345.5\frac{d^4}{N} + d_1^- - d_1^+ = 1$ $0.0191\frac{1}{d^2(N+2)} - d_2^- + d_2^+ = 1$ *Bounds on System Variables:* $N \geq 3.5$ $0.059 \leq d \leq 0.09$ **Minimize** *Deviation Function:* $Z = w_1 d_1^- + w_2 d_2^+$

Currently, we standardize the structure of product information according to a set of XML schemas (or templates). XML offers a convenient and standardized means of capturing information at the product information level and ensures that problem specific *declarative* information can be reused in different processes. For the simple example problem of designing a pressure vessel and a spring, the product information is stored in four XML templates: the problem definition template, the constraints template, the goals and preferences template, and the analysis code template. These templates, discussed next, correspond to the declarative product information "hidden" within the compromise DSP formulation shown in Table 2.

cDSP "Chips"	Pressure Vessel		Spring	
Constraints	Stress: $\frac{PR}{T} - S_t \le 0$ Thickness: $5T - R \le 0$ Radius: $R + T - 40 \le 0$ Length: $L + 2R + 2T - 150 \le 0$		Minimum Deflection: $\delta = \frac{8FD^3N}{d^4G} \ge 1.1$ Maximum Solid Height: $H = N_t \cdot d \le 0.5$	
Variables	Radius, Length, Thickness		Number of Coils, Wire Diameter	
Parameters	Density, Strength, Pressure		Applied Force, Coil Diameter, Shear Modulus	
Goals	Volume Target = 500000 m^3 Weight Target = 300 kg		Stiffness Target = lbf/in Volume Target = in^3	
Preferences	Volume Weighting Factor = 0.5 Weight Weighting Factor = 0.5		Stiffness Weighting Factor = 0.5 Volume Weighting Factor = 0.5	
Objective	**Maximize** Volume $V(R,L) = \pi\left[\frac{4}{3}R^3 + R^2L\right]$ **Minimize** Weight $W(R,T,L) = \pi\rho\left[\frac{4}{3}(R+T)^3 + (R+T)^2L - \left(\frac{4}{3}R^3 + R^2L\right)\right]$		**Maximize** Stiffness $k = \frac{d^4G}{8D^3N}$ **Minimize** Volume $V = \frac{1}{4}\pi^2 Dd^2(N+2)$	
Analysis	**Inputs**	**Outputs**	**Inputs**	**Outputs**
	Radius, Length, Thickness, Density, Strength, Pressure	Volume, Weight	Wire Diameter, Coil Diameter, Shear Modulus, Number of Coils	Stiffness, Volume
Driver	**Optimization Algorithm** – Exhaustive Search, SQP, etc.		**Optimization Algorithm** – Exhaustive Search, SQP, etc.	
Response	**Design Variable Values** – Radius, Length, Thickness **Objective Function Value** - Z		**Design Variable Values** – Radius, Length, Thickness **Objective Function Value** - Z	

Figure 6. Product information level (declarative product level) for pressure vessel and spring design.

1. Variables and Parameters Definition Template

The template for defining design variables and parameters includes the following information about design variables: a) Design variable Name b) Type c) Unit d) Value e) Lower bound and Upper bound. The parameters are defined with equal lower and upper bounds. The XML schema representation associated with the problem definition template is shown in Figure 8.

2. Constraints Definition Template

The constraints definition template includes information about various constraints on the system. The constraints are associated with a name and a string representing required mathematical operations. The XML schema representation associated with the constraints definition template is provided in Figure 8.

3. Goals and Preferences Definition Template

In this template, information about design goals and designer preferences regarding the satisfaction of these goals is captured. The goals are formulated with target values for system responses. Preferences are associated with the various goals included in the compromise DSP formulation. Here, these preferences are

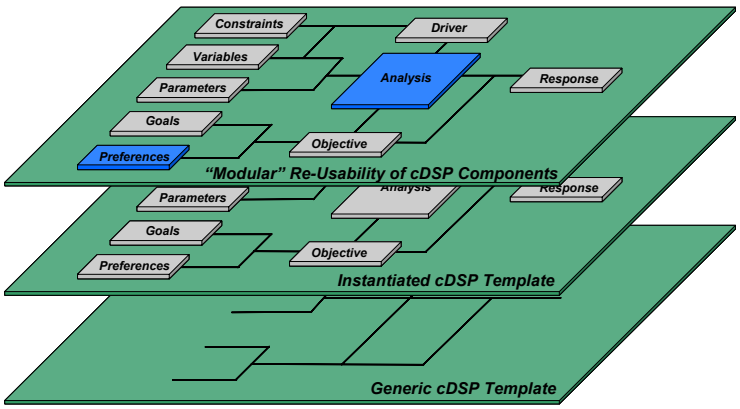


Figure 7 - Archival, Documentation, and Re-use of Design Process Building Blocks

modeled as weights on the deviation variables. The entities associated with goals are: a) Name b) Weight c) Target and d) Monotonicity. Monotonicity captures information regarding whether the goal is to be maximized, minimized, or matched as closely as possible. The XML schema associated with the goals and preference definition templates is shown in Figure 8.

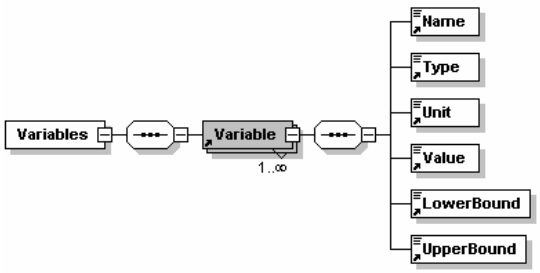
4. Analysis Code Template

The analysis code is used to evaluate the system response to changes in design variables. The information associated with the analysis code template includes a) Inputs, which consist of Name, Type, Unit, and Value, b) Outputs, which consist of: Name, Type, Unit, and Value and c) Execute. The "Execute" field captures the software application that needs to be executed in order to obtain the system response. The XML schema associated with the analysis code template is shown in Figure 8.

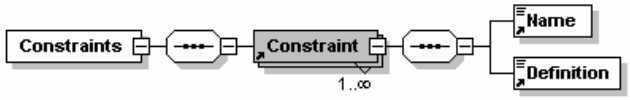
B. Process Level (Declarative Process Level)

In the layer, corresponding to the Process Level (1) required information transformations are identified and (2) required information flows are specified accordingly. In order to ensure that declarative information is separated from procedural information, information flows are clearly separated from information content. In other words, we capture only the mechanics of information transfer at this level, while problem specific information is defined separately at the declarative level. This results in a process map that remains the same irrespective of the application in which the process is used. Information content is thus effectively batched, according to the structure of the overarching template.

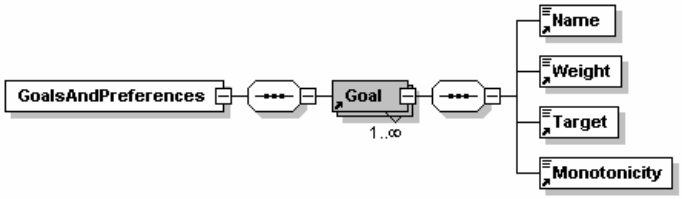
A simple example of a generic process map for the design of either a spring or a pressure vessel using compromise DSP construct, discussed in Section III, is given in Figure 9. The elements of this generic process include problem definition, analysis, constraint evaluation, goals evaluation, and an optimization routine. Each of these entities interacts with the product information layer through the product information templates discussed in Section IV.A. The information flows between these entities are generic and independent of the product being designed. For example, the flow of information between the analysis module and constraints evaluation include the problem name, an array of input names (i.e., design variables) and an array of input values. The actual input names and values are dependent on the problem and are extracted from the



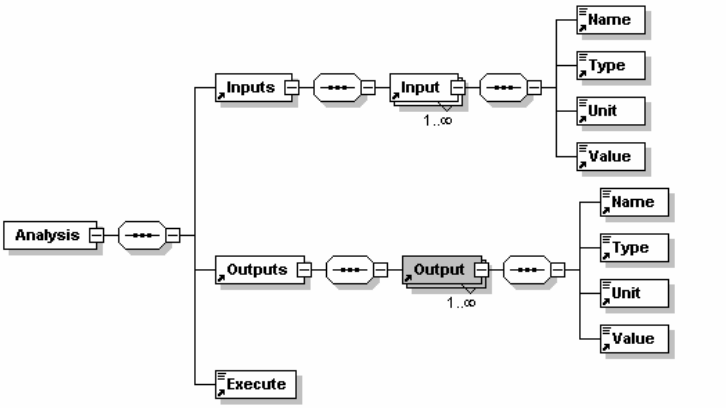
Schema representation for problem definition



Schema representation for constraints



Schema representation for goals and preferences



Schema representation for analysis code

Figure 8 – Schemas for product information

variables and parameter definition template discussed in Section IV.A.1.

The implementation of the declarative process level relies on the use of ModelCenter®[36] developed by Phoenix Integration Inc. ModelCenter® allows modeling design processes in terms of various simulation codes and the information required to flow among them. Associated with each entity in this process are a set of JavaBeans that parse information from appropriate XML files at the product information level and make information available for processing in ModelCenter®. These Process elements are mapped to each other for a specific problem, in a manner that reflects the underlying (batched) information flows required by the generic templates. This mapping remains the same irrespective of the design problem in which the process is used. For example, the information flows and mappings relevant for the solution of a compromise DSP, will remain the same, whether we are designing a pressure vessel or a spring.

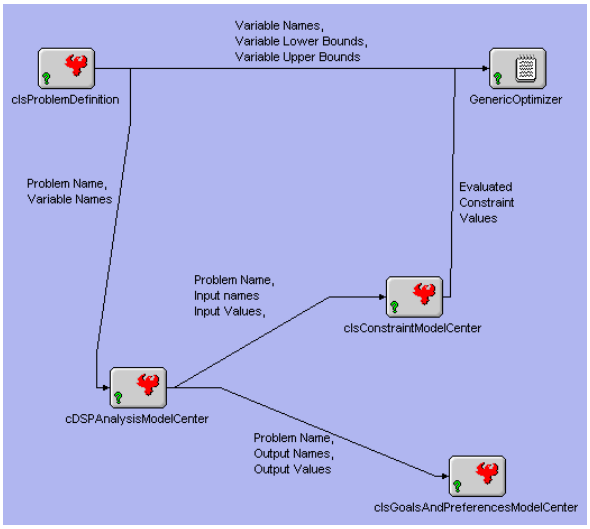


Figure 9 - Process map for design of spring / pressure vessel

C. Execution Level (Procedural Level)

The details of code execution are captured in the layer, corresponding to the Execution Level. This level is specific to the design problem for which the process is used. Execution level codes interface only with the declarative problem formulation level. Thus, there is no direct link between the process specification level and the execution level. This architecture preserves the modularity of the design processes being modeled. For the design of the pressure vessel and the spring, the execution level codes (i.e., the analysis codes for both the pressure vessel and the spring) have been written in Visual Basic, although any other model wrapped as a ModelCenter® component could be used.

The results obtained for the pressure vessel and spring design, using the generic process, pictured in Figure 9, are summarized in Table 3 and Table 4, respectively. These results have been verified and validated with exhaustive searches, based on more traditional problem formulations.

Table 3 - Results for pressure vessel problem

Design Variable	Value
Radius (R)	10 mm
Length (L)	80 mm
Thickness (T)	3.5 mm
Objective function (Z)	0.497

Table 4 - Results for spring design problem

Design Variable	Value
Coil Diameter (d)	0.059 in
Number of Coils (N)	3.5
Objective function (Z)	0.655

V. Design Process Reusability: Composition of Templates

Having illustrated our approach for the simple design scenarios involved in spring and pressure vessel design, we now shift our attention to a more complex (e.g., involving further complications due to distribution, complexity, collaboration, multi-functionality, etc.) design example, namely that of designing Linear Cellular Alloys (LCAs). Specifically, we extend the templates, discussed in Section IV, to model the required design process, thereby illustrating the ability to compose design process templates from existing sub-process templates.

Linear Cellular Alloys are honeycomb materials (see Figure 10) that are processed through the extrusion of slurry through a multistage die. The slurry is composed of a binder mixed with metal oxide powders. The structure resulting from extrusion is first dried and reduced into the metallic phase in a hydrogen rich environment and then sintered to achieve nearly fully dense metal composites. A wide range of cell sizes and shapes including functionally graded structures can be achieved using this manufacturing process. These materials are suitable for multi-functional applications that require both strength and heat transfer capabilities[37]. Applications of these materials include heat sinks for microprocessors and combustor liners for aircraft engines. One of the main advantages of these LCAs is that desired structural and thermal properties can be obtained by designing the shape, arrangement of cells, selecting appropriate cell wall thicknesses and setting appropriate dimensions for the LCA.

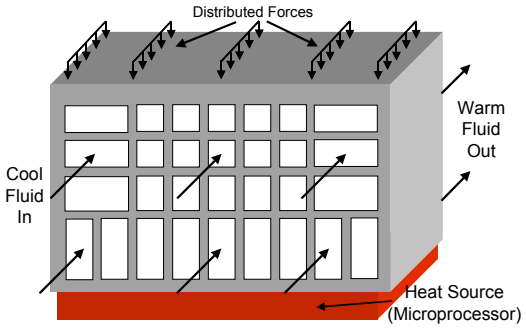


Figure 10. Linear Cellular Alloys with rectangular cells.

Assuming the construction of a structural heat exchanger, using LCAs, the required design process must encompass both structural and thermal considerations. One way of structuring this design process is as a series of information transformations that reduce the available design space according to the perspectives considered. In such a sequential design process, designers consider sets of design alternatives rather than pursuing a particular alternative directly, as shown in Figure 11. The underlying philosophy is to gradually concretize the design space until a final solution is achieved. In the LCA design scenario, considered, this approach may be implemented as one designer (thermal or structural) generating a range of design parameters and subsequently passing on this range to the next designer to select the best value within the proposed range. Therefore, the process of designing multifunctional LCAs involves two compromise DSPs, one for structural and one for thermal considerations, respectively. Each of these compromise DSPs is solved using an independent instantiation of the process shown in Figure 9. This process (see Figure 9) serves as a reusable building block and thus constitutes sub-process of the overall LCA design process. Here, again, the declarative, process specific information remains unchanged. Since only the product specific information must be altered or adapted for application in modeling the LCA design problem, reusability of the process is enhanced.

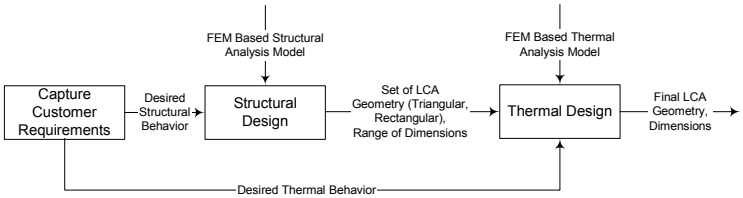


Figure 11. Process of Design of LCAs.

Although the LCAs pertain to an emerging class of multifunctional structure-material systems that often span multiple functional domains as well as length/time scales, the underlying design process is clearly decomposable. Different designers contribute their expertise to each aspect of the multifunctional design. There are a number of inherent benefits to the proposed modeling technique for design processes. On the whole, there are three main functions, namely *computer interpretability*, *modularity*, and *archival*, that each in turn allow for the execution, re-use/reconfiguration, and documentation of design processes and any of their components, respectively. The fundamental methodological differences between the proposed design process modeling technique and other commonly available approaches are that (1) declarative and procedural models are effectively separated and (2) the process elements are composable as modular building blocks. Consequently, it is possible to effectively model design processes and sub-processes, regardless of functional domain or complexity.

VI. Closure

In this paper, we present an approach for modeling design processes, based on a modular, decision-centric, template-based representation of design processes. These process templates are computer interpretable and archivable, allowing for the execution, re-use/reconfiguration, and documentation of design processes and any of their components, respectively. A modular, generic formulation of the process required for the solution of a compromise DSP is conceived and presented in Section IV. Furthermore, the underlying information model is formalized and the resulting construct is implemented in ModelCenter®. Finally, the developed process model is

instantiated and validated for two examples, namely, pressure vessel and spring design, in order to offer proof of concept for the proposed modeling technique. Reusability of the process model for solving compromise DSPs as sub-process is then illustrated for the design of Linear Cellular Alloys.

There are a number of changes that can take place with regard to any given design process. The intent in the proposed modeling approach is to isolate the effects of each of these changes by providing the required level of modularity. The principal advantage of this approach is the enhanced reusability of information and knowledge achieved via the separation of information pertaining to problem formulation, process, and execution. However, we recognize that the value and ease of implementing a template-based design approach increases with the quantity and quality of information available. Thus, while it is possible to formulate templates, at least structurally, even in the early stages of design, the information and knowledge gained by exercising the resulting models becomes more concrete as the design matures. The advantage of relying on completely modular templates is the provision of a consistent means of capturing and exploiting knowledge that reflects evolving information content throughout the design process.

The overarching goal of this research is to formalize a declarative design process modeling technique, centered on decision-centric design processes. We have already implemented compromise DSPs as modular, reusable, template-based design process building blocks, taking advantage of the consistent, application independent structure of this construct. Future efforts will center on formalizing the remaining information transformations (e.g., abstraction, composition, decomposition, interface, and mapping), mentioned in Section III, in an analogous fashion. This requires: *(1)* mapping information schemas defined at various levels of abstraction, *(2)* developing a design process repository *(3)* developing metrics for characterizing individual information transformations and their compositions, *(4)* formalizing interactions among the stakeholders involved in a shared design effort, *(5)* exploring design process architecture and development of design process families, and *(6)* investigating effects of stakeholder control on design processes and value chain modularity. We believe that this research is a stepping stone towards top-down design of design processes from existing design process knowledge. The existence of a repository of design process building blocks will greatly facilitate the original, adaptive, derivative, and variant design of products and serves as a springboard for the evolution of a product portfolio.

Acknowledgements

We acknowledge the support from National Science Foundation grants DMI-0085136 and DMI-0100123, as well as, Air Force Office of Scientific Research grant F49620-03-1-0348. Marco Gero Fernández is sponsored by a National Science Foundation IGERT Fellowship through the TI:GER Program at the Georgia Tech College of Management and a President's Fellowship from the Georgia Institute of Technology.

References

[1]Pahl, G. and W. Beitz, 1996, *Engineering Design - A Systematic Approach,* Springer, Berlin, Heidelberg, New York.

[2]Moder, J. J., C. R. Phillips and E. W. Davis, 1983, *Project management with CPM, PERT, and precedence diagramming,* New York : Van Nostrand Reinhold.

[3]Modell, M. E., 1996, ***A Professional's Guide to Systems Analysis,*** McGraw-Hill Book Company, New York, NY.

[4]1993, *Integrated Definition for Functional Modeling (IDEF 0)*, Federal Information Processing Standards Publication 183.

[5]Park, H. and M. R. Cutkosky, 1999, "Framework for Modeling Dependencies in Collaborative Engineering Process," *Research in Engineering Design,* Vol. 11, No. 2, pp. 84-102.

[6]Whitney, D. E., 1996, "Why Mechanical Design Cannot be Like VLSI Design," *Research in Engineering Design,* Vol. 8, No. 3, pp. 125-138.

[7]Eppinger, S., D. E. Whitney, R. P. Smith and D. A. Gebala, 1994, "A Model-based Method for Organizing Tasks in Product Development," *Research in Engineering Design,* Vol. 6, No. 1, pp. 1-13.

[8]Shimomura, Y., M. Yoshioka, H. Takeda, Y. Umeda and T. Tomiyama, 1998, "Representation of Design Object Based on the Functional Evolution Process Model," *Journal of Mechanical Design,* Vol. 120, No. 2, pp. 221-229.

[9]Ullman, D. G., 1992, "A Taxonomy for Mechanical Design," *Research in Engineering Design,* Vol. 3, No. 3, pp. 179-189.

[10]Maimon, O. and D. Braha, 1996, "On the Complexity of the Design Synthesis Problem," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans,* Vol. 26, No. 1, pp. 142-151.

[11]Maher, M. L.,1990, *Process Models for Design Synthesis*. AI Magazine. **Winter 1990:** 49-58.

[12]Gorti, S. R., A. Gupta, G. J. Kim, R. D. Sriram and A. Wong, 1998, "An Object-Oriented Representation for Product and Design Process," *Computer Aided Design,* Vol. 30, No. 7, pp. 489-501.

[13]Muster, D. and F. Mistree, 1988, "The Decision Support Problem Technique in Engineering Design," *International Journal of Applied Engineering Education,* Vol. 4, No. 1, pp. 23-33.

[14]Elmaghraby, S. E., 1995, "Activity Nets: A Guided Tour Through Some Recent Developments," *European Journal of Operational Research,* Vol. 82, No. 3, pp. 383-408.
[15]Browning, T. R. and S. D. Eppinger, 2002, "Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development," *IEEE TRansactions on Engineering Management,* Vol. 49, No. 4, pp. 428-442.
[16]Umeda, Y., H. Takeda, T. Tomiyama and H. Yoshikawa, 1990, "Function, Behavior, and Structure," *AIENG '90 Applications of AI in Engineering*, Computational Mechanics Publications and Springer Verlag**,** pp. 177-193.
[17]Hsu, Y.-L., C.-Y. Tai and Y.-C. Chen, 2000, "A Design Process Model Based on Product States," *Journal of Chinese Society of Mechanical Engineers,* Vol. 21, No. 4, pp. 369-377.
[18]Tomiyama, T. and H. Yoshikawa, 1986, "Extended General Design Theory," *Design Theory for CAD, Proceedings of the IFIP WG5.2 Working Conference 1985*, Elsevier, North-Holland, Amsterdam, pp. 95-130.
[19]Takeda, H., P. Veerkamp, T. Tomiyama and H. Yoshikawa,1990, *Modeling Design Processes*. AI Magazine. **Winter 1990:** 37-48.
[20]Zeng, Y. and P. Gu, 1999, "A Science Based Approach to Product Design Theory Part 1: Formulation and formalization of Design Process," *Robotics and Computer Integrated Manufacturing,* Vol. 15, No. 6, pp. 331-339.
[21]Mistree, F., B. A. Bras, W. F. Smith and J. K. Allen, 1996, "Modeling Design Processes: A Conceptual, Decision-Based Perspective," *International Journal of Engineering Design and Automation,* Vol. 1, No. 4, pp. 209-221.
[22]Mistree, F., W. F. Smith, B. Bras, J. K. Allen and D. Muster, 1990, "Decision-Based Design: A Contemporary Paradigm for Ship Design," *Transactions, Society of Naval Architects and Marine Engineers*, Jersey City, New Jersey, Vol. 98**,** pp. 565-597.
[23]Mistree, F., W. F. Smith and B. A. Bras, 1993, "A Decision-Based Approach to Concurrent Engineering," *Handbook of Concurrent Engineering* (H. R. Paresai and W. Sullivan, Eds.), Chapman & Hall, New York**,** pp. 127-158.
[24]Mistree, F., W. F. Smith, S. Z. Kamal and B. A. Bras, 1991, "Designing Decisions: Axioms, Models and Marine Applications," *Fourth International Marine Systems Design Conference*, Kobe, Japan, Society of Naval Architects of Japan, pp. 1-24.
[25]Bras, B., Mistree, F., 1991, "Designing Design Processes in Decision-Based Concurrent Engineering," *SAE Transactions Journal of Materials & Manufacturing*, pp. 451-458.
[26]Mistree, F., K. Lewis and L. Stonis, 1994, "Selection in the Conceptual Design of Aircraft," *5th AIAA/USAF/NASA/ISSMO Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*, Panama City, FL, pp. 1153-1166, pp. 1153-1166.
[27]Fernández, M. G., C. C. Seepersad, D. W. Rosen, J. K. Allen and F. Mistree, 2001, "Utility-Based Decision, Support for Selection in Engineering Design," *13th Internation Conference on Design Theory and Methodology*, Pittsburgh, PA. Paper Number: DETC2001/DAC-21106.
[28]Mistree, F., O. F. Hughes and B. A. Bras, 1993, "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm," *Structural Optimization: Status and Promise* (M. P. Kamat, Ed.), AIAA, Washington, D.C.**,** pp. 247-286.
[29]Seepersad, C. C., F. Mistree and J. K. Allen, 2002, "A Quantitative Approach for Designing Multiple Product Platforms for an Evolving Portfolio of Products," *ASME Design Engineering Technical Conferences, Advances in Design Automation*, Montreal, Canada. Paper Number: DETC2002/DAC-34096.
[30]Schlenoff, C., A. Knutilla and S. Ray,1996, *Unified Process Specification Language: Requirements for Modeling Process*. Gaithersburg, MD, National Institute of Standards and Technology.
[31]Mistree, F., D. Muster, J. A. Shupe and J. K. Allen, 1989, "A Decision-Based Perspective for the Design of Methods for Systems Design," *Recent Experiences in Multidisciplinary Analysis and Optimization*, Hampton, Virginia. Paper Number: NASA CP 3031.
[32]Kamal, S. Z., H. M. Karandikar, F. Mistree and D. Muster, 1987, "Knowledge Representation for Discipline-Independent Decision Making," *Expert Systems in Computer-Aided Design* (J. Gero, Ed.), Elsevier Science Publishers B.V., Amsterdam**,** pp. 289-321.
[33]Pahl, G. and W. Beitz, 1996, *Engineering Design: A Systematic Approach,* Springer-Verlag, New York.
[34]Simon, H. A., 1996, *The Sciences of the Artificial,* MIT Press, Cambridge, Mass.
[35]Simon, H. A.,1976, *Administrative Behavior*. New York, N.Y., The Free Press.
[36]Phoenix Integration Inc.,2004, *ModelCenter®, Version 5.0*.
[37]Seepersad, C. C., B. M. Dempsey, J. K. Allen, F. Mistree and D. L. McDowell, 2002, "Design of Multifunctional Honeycomb Materials," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA. Paper Number: AIAA-2002-5626.